STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVI**, Number 2, 2011

ONTOLOGY ENRICHMENT USING SEMANTIC WIKIS AND DESIGN PATTERNS

MARIUS GEORGIU AND ADRIAN GROZA

Abstract. This research addresses the task of ontology enrichment by exploiting the large amount of structured information available in semantic wikis. The proposed solution makes use of ontological design patterns to guide the semiautomatic enrichment process and regular expressions or predefined values in the automatic enrichment process.

1. Introduction

Ontology enrichment generates extensions, in terms of new concepts, new relations, or corrections of the axioms of an existing ontology. In semi-automatic settings, these extensions are proposed to the ontology engineers for assessment, whilst in an automatic setting, the ontology is enriched based on algorithms that add, remove, or update terms from the ontology. The potential of combining Web 2.0 with Web 3.0 is advocated in literature [1]. At the moment, we are at the beginning of developing the social computing science [6]. In this line, the current study applies the active social machine behind semantic wikis to the hard task of ontology maintenance.

2. Technical instrumentation

Semantic wikis provide users the capability to annotate their text with specific concepts and roles from a set of imported ontologies, in order to be processed against semantic queries. Among the available semantic wikis, such as DBpedia [3], ACEWiki [7], or OntoWiki [2], we drive our attention towards Semantic Media Wiki (SMW), due to its success in terms of number of users. The structure of an annotation in SMW *[[property::value]]* has the role to associate a value to a property. The properties are defined by pages which correspond to the *Property* namespace, where one can find information

Received by the editors: April 10, 2011.

2000 *Mathematics Subject Classification.* 68T05.

1998 *CR Categories and Descriptors.* I.2.6 [**Artificial Intelligence**]: Learning – *Knowledge acquisition.*

Key words and phrases. ontology enrichment, semantic wikis, ontology design patterns.

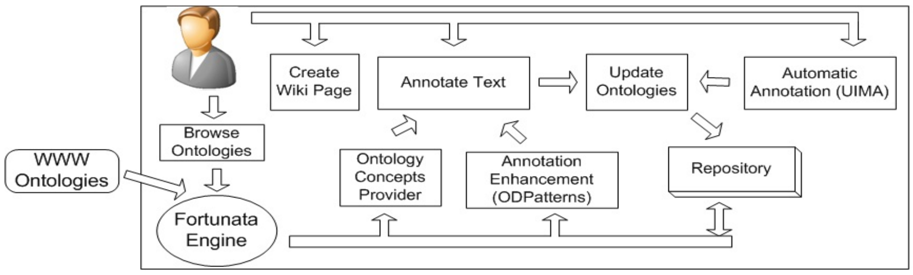

FIGURE 1. System Architecture.

regarding how to use that property. A semantic query consists of two parts: the first one defines the pages where information is searched and the second part defines the searched information. The query [[$Memory : DDR3$]] returns all pages with *Memory* property having *DDR3* value.

Ontology design patterns (ODPs) offer solutions to a series of issues which appear recurently when developing an ontology. The solution is oriented towards providing guidance related to a collection of questions: "which agent does play this role?" as in the *Agent Role* ODP. The ODPs used here belong to several categories: i) classification, to represent the relations between concepts and entities which concepts can be assigned to, ii) collection, to represent domain membership, and price, to represent the price for different objects.

3. System Architecture

In our approach the ontology is enriched with terms taken from semantic wikis. In the first step users annotate documents based on the imported ontologies in the system. In the second step the initial ontology is enriched based on these documents. Consequently, new wiki pages would be annotated based on an up-to-date ontology. The ontology enrichment process is guided by ontology design patterns and heuristics such as the number of annotations based on a concept or an instance.

The system relying on pipe-based architecture is able to extract semantic information in an automatic manner from Wiki pages and use it in order to improve an existing ontology. The role of each component in figure 1 follows: i) *Automatic Annotation* (UIMA) extracts and annotates text automatically from Wiki pages; ii) *Annotate Text* annotates the text with information from user; iii) *Create Wiki Page* creates a new Wiki page; iv) *Update Ontologies* adds new terms into system ontology found after annotation process; v) *Ontology Concepts Provider* displays all concepts from system's ontology when the user makes an annotation; vi) *Annotation Enhancement* (ODPatterns)

1. $Laptop \sqsubseteq Computer$
2. $Computer \sqsubseteq \exists part.Processor \sqcap \exists part.Memory$
3. $compaq6710s : Laptop$
4. $intelCore2Duo : Processor$
5. $ddr3 : Memory$

Figure 2. Part of the initial ontology

helps the user in the annotation process by suggesting the rest of terms to be annotated from an ODP when the user annotates a term from that ODP; vii) *Fortunata Engine*, the core component which manages and integrates the other modules; viii) *Repository*, a database where wiki pages and ontologies are stored; ix) *Browse Ontologies*, displays ontologies in a human readable format; x) *WWW Ontologies*, ontologies imported in Fortunata. The system is built on top of the Fortunata engine to integrate reasoning with ODP. Fortunata is an extensible Semantic Web tool which allows developers to implement plugins integrated with ontologies that can be presented in a human readable format.

The automatic annotation is based on the UIMA (Unstructured Information Management) framework, main task here being to provide the right input to UIMA which annotates the text automatically. The annotation process is based on regular expressions or predefined values which match certain words from a text. The regular expression which identifies prices from a text consisting in a number and a currency follows

```
<name>Patterns</name>
<value><array><string>[0-9]+[ ]*(?i)(ron|euro)</string></array></value>
```

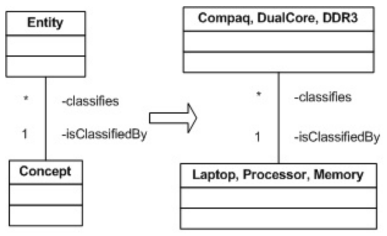

Figure 3. Instances of the classification design pattern

4. Running Scenario

As a proof of concept we consider the dynamic domain of Information Technology (IT), where new concepts are upgraded versions of the old ones. In order to increase the matching between clients requests or searches and its own offer, an IT store decides to describe its products according to the vocabulary of the potential consumers. The employees can annotate information about

```
<typeDescription>
    <name>ro.utcluj.uima.ontology.concept.Price</name>
    <description></description>
    <supertypeName>uima.tcas.Annotation</supertypeName>
    <features><featureDescription>
                  <name>hasValue</name>
                  <description>Value</description>
                  <rangeTypeName>uima.cas.String</rangeTypeName></featureDescription>
              <featureDescription>
                  <name>hasCurrency</name>
                  <description>Currency</description>
                  <rangeTypeName>uima.cas.String</rangeTypeName></featureDescription></features>
</typeDescription>
```

FIGURE 4. Generated XML from Price ODP.

their products and use it for semantic queries. To accomplish this, its own IT ontology will be updated according the definitions encountered on wikipedia.

The main steps of the semi-automatic process performed by an employee and the system when enriching the ontology start by loading the innitial ontology (figure 2). A *Laptop* has one or more processors and one or more memories (axiom 2), whilst $compaq6710s$ is an instance of *Laptop* (axiom 3), $intelCore2Duo$ an instance of $Processor$ (axiom 4) and $DDR3$ an instance of $Memory$ (axiom 5). For storing a new model of laptop $Lenovo560$ one has to create a new Wiki page containing the text: *"Lenovo560 has been added into our store at only 800 euro"*. The selected text *Lenovo560* is annotated as being a *Laptop* by selecting this concept from the list of available concepts. Further, the ontology enrichment process is launched based on ODPs. For the $classification$ design pattern the new model of laptop has been already classified by user's manually annotation: $Lenovo560$ is an instance of *Laptop* as displayed in figure 3. Mapping this pattern over IT domain, we obtain in the right side of the figure certain concepts and instances. For the *price* design pattern the system asks user about laptop's price and currency. The ontology of the system is enriched with a new instance of *Laptop* and its price. After importing it, the system extracts classes and their properties into XML files (see figurefigure 4) and generates further Java classes used by UIMA. Here, *typeDescription* describes the beginning of a new type/class generated by UIMA; *name*, fully qualified name of the new class; $supertypeName$, fully qualified name of the new supper class; $features$ presents the list of all attributes from the new class; $featureDescription$ marks the beginning of an attribute from the new class; *name*, name of the attribute; $rangeTypeName$, type of the attribute.

UIMA uses an annotator configured with predefined regular expression for new generated Java classes to annotate the text automatically, as follows: *"Found Price instance: begin=48 end=56 hasValue=800 hasCurrency=euro"*.

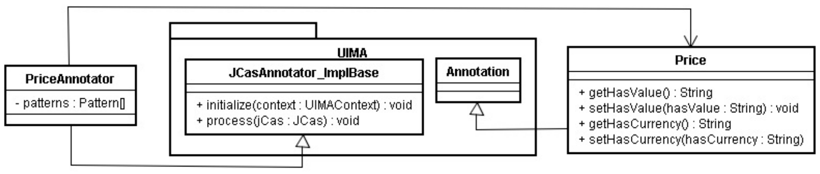
UIMA
PriceAnnotator
- patterns : Pattern[]
JCasAnnotator_ImplBase
+ initialize(context : UIMAContext) : void
+ process(jCas : JCas) : void
Annotation
Price
+ getHasValue() : String
+ setHasValue(hasValue : String) : void
+ getHasCurrency() : String
+ setHasCurrency(hasCurrency : String)

Figure 5. Diagram Classes for Price Annotation.

One can see the start and end indexes of the instance within the text marked with *begin*, *end* tokens and the values for *hasValue* and *hasCurrency* attributes extracted from *Price* ODP into Java classes displayed in figure 5. *PriceAnntator* represents the class which processes the text and creates *Price* annotations.

5. Discussions and Related Work

Information extraction methods are applied in the context of semantic wikis in [9]. A semi-automatic solution to enrich ontologies [5] performs several transformation operations against the content of Web pages: searcing for words that appear frequently in the pages, classifying these words based on heuristics, extracting ontology elements and revising the final ontology by a human expert that can make modifications against the result.

Design patterns are used in [8] to manage ontologies in an easier manner. ODPs are building blocks for ontology management representing small ontologies that can be extended and adapted to a specific application. An initial ontology is enriched based on ODPs in [4]. The process is semi-automatic and has been implemented in two phases: i) element extraction - uses an initial ontology in order to extract elements together with a confidence ii) patterns matching and ranking - evaluates against ODPs the ontology elements previously extracted based on words metrics or using WordNet. The ontology is evaluated and enriched with the best new elements. In our case, the up to date knowledge is extracted from semantic wikis in a solution that enriches automatically system ontology using regular expressions or predefined values for ontology elements extraction.

6. Conclusion

The main contribution here has been to exploit design patterns to structure the automatic ontology enrichment process using semantic wikis. Ongoing work regards the assessment of the proposed methodology against ontology evaluation metrics based on features such as inheritance richness, attribute richness, and class richness, within an ontology evaluation framework [10].

Acknowledgment

We are grateful to the anonymous reviewers for their useful comments. This work was supported by the grant ID 160/672 from the National Research Council of Romanian Ministry of Education and Research and POS-DRU/89/1.5/S/62557/EXCEL.

References

1. Anupriya Ankolekar, Markus Krtzsch, Thanh Tran, and Denny Vrandecic, *The two cultures: Mashing up web 2.0 and the semantic web*, Web Semantics: Science, Services and Agents on the World Wide Web **6** (2008), no. 1, 70 – 75, Semantic Web and Web 2.0.
2. Sören Auer, Sebastian Dietzold, and Thomas Riechert, *Ontowiki - a tool for social, semantic collaboration*, International Semantic Web Conference, 2006, pp. 736–749.
3. Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann, *Dbpedia - a crystallization point for the web of data*, J. Web Sem. **7** (2009), no. 3, 154–165.
4. Eva Blomqvist, *Ontocase-automatic ontology enrichment based on ontology design patterns*, Proceedings of the 8th International Semantic Web Conference (Berlin, Heidelberg), ISWC '09, Springer-Verlag, 2009, pp. 65–80.
5. Timon C. Du, Feng Li, and Irwin King, *Managing knowledge on the web - extracting ontology from html web*, Decision Support Systems **47** (2009), no. 4, 319–331.
6. Jim Hendler and Tim Berners-Lee, *From the semantic web to social machines: A research challenge for ai on the world wide web*, Artif. Intel. **174** (2010), no. 2, 156 – 161.
7. Tobias Kuhn, *Acewiki: Collaborative ontology management in controlled natural language*, SemWiki, 2008.
8. Valentina Presutti and Aldo Gangemi, *Content ontology design patterns as practical building blocks for web ontologies*, ER, 2008, pp. 128–141.
9. Pavel Smrz and Marek Schmidt, *Information extraction in semantic wikis*, SemWiki (Christoph Lange 0002, Sebastian Schaffert, Hala Skaf-Molli, and Max Völkel, eds.), CEUR Workshop Proceedings, vol. 464, CEUR-WS.org, 2009.
10. Samir Tartir and Ismailcem Budak Arpinar, *Ontology evaluation and ranking using ontoqa*, ICSC, 2007, pp. 185–192.

Technical University of Cluj-Napoca, Department of Computer Science, Baritiu 28, RO-400391 Cluj-Napoca, Romania

E-mail address: `mariusgeorgiu@yahoo.com, adrian.groza@cs-gw.utcluj.ro`